INFORMATION AND CONTROL 26, 24–33 (1974)

Adjacency in Digital Pictures*

AZRIEL ROSENFELD

Computer Science Center, University of Maryland, College Park, Maryland 20742

Let S be a subset of a digital picture, and let $\bar{S}$ be the complement of S. It is well known that the connected components of S and $\bar{S}$, under the relation "is adjacent to," form a tree, and algorithms for constructing this tree have been devised. The main purpose of this paper is to prove that the components do form a tree, and in the process, to provide a basis for proving the validity of the tree-constructing algorithms.

1. BASIC CONCEPTS

Let $I \times I$ be the set of pairs of integers, and let $x = (i, j)$ be in $I \times I$. The four pairs $(i \pm 1, j)$ and $(i, j \pm 1)$ are called the 4-*neighbors* of x, and are said to be 4-*adjacent* to x. These same pairs, together with the four pairs $(i \pm 1, j \pm 1)$, are called the 8-neighbors of x (8-*adjacent* to x). More generally, if S, T are subsets of $I \times I$, we say that S is (4- or 8-) adjacent to T if some $x \in S$ is (4- or 8-) adjacent to some $y \in T$. From now on, elements of $I \times I$ will be called *points*.

A *path* from x to y is a sequence of distinct points $x = x_0, x_1, \ldots, x_{n-1}, x_n = y$ such that x_i is adjacent to x_{i-1}, $1 \leqslant i \leqslant n$. There are two versions of this definition, "4-path" and "8-path", depending on which type of adjacency is used. Similarly, there are two versions of most of the definitions which follow.

If $x, y \in S \subseteq I \times I$, we say that x is *connected* to y in S if there is a path from x to y consisting entirely of points of S. Readily, "is connected to" is an equivalence relation; the equivalence classes under this relation are called the connected *components* of S. If S has only one component, it is called *connected.*

PROPOSITION 1. *If S contains T, every component of T is contained in a unique component of S.*

* The support of the Office of Naval Research, under Contract N00014-67-A-0239-0012, is gratefully acknowledged. The author is also indebted to Anupam N. Shah for several helpful discussions, and to Mrs. Eleanor B. Waters for her help in preparing the manuscript.

Let $\bar{S}$ be the complement of S. If S is finite, all points of $\bar{S}$ that are sufficiently far from the origin are evidently connected in $\bar{S}$. It follows that exactly one component of $\bar{S}$ is infinite; we call this component the *background* of S. All other components of $\bar{S}$, if any, are called *holes* in S. If S is connected and has no holes, it is called *simply-connected.* We assume from now on that S is finite. We shall sometimes refer to points of S as 1's, and to points of $\bar{S}$ as 0's; this convention is suggested by the definition of the "characteristic function" K_S of a set S: $K_S(x) = 1$ if $x \in S$, $K_S(x) = 0$ if $x \in \bar{S}$.

Many of the theorems that can be proved about connectedness hold only if opposite definitions are used for S and $\bar{S}$; i.e., if we use 4-connectedness for 1's and 8-connectedness for 0's, or vice versa. We shall do so in the remainder of this paper. Thus, for example, we now have only two types of simply-connected S's: a 4-connected S that has no 8-holes, and an 8-connected S that has no 4-holes.

PROPOSITION 2. *If $\bar{S}$ is connected, every component of S is simply-connected. If $\bar{S}$ has $k + 1$ components, no component of S has more than k holes.*

It should be pointed out that if we use a hexagonal grid of points in place of $I \times I$, all of the results of this paper remain valid, with the simplification that only one type ("6-") of adjacency, connectedness, etc., is needed. Generalizations to other tessellations of the plane are also possible, but would seem to be of little practical interest. For surfaces more complicated than the plane, e.g., surfaces such as the torus, many of our results are no longer valid.

2. BORDERS

By the *border* of S we mean the set of points of S that are adjacent to points of $\bar{S}$. More generally, if C is a component of S, and D is a component of $\bar{S}$, by the D-border of C we mean the set of points of C that are adjacent to points of D. From now on, we shall use only the versions of these definitions in which "adjacent" means "4-adjacent".

It is known (Rosenfeld, 1970) that if S is simply-connected, a certain algorithm will find all border points of S, given an initial border point. In the following paragraphs we state a slightly modified version of this algorithm and prove that, for any C and D, it can be used to find all points of the D-border of C, given an initial pair of 4-adjacent points c, d with $c \in C$, $d \in D$. We assume below that C has more than one point; otherwise its sole point c is also its D-border, and there is nothing to do.

Our algorithm, which we shall refer to as BF, proceeds as follows.

(a) Change the values of c and d to 3 and 2, respectively.

(b) Let the 8-neighbors of c in (say) clockwise order, starting with d and ending with the first occurrence of 1, 3, or 4, be $e_1, \ldots, e_k$. If $c = 3$, $e_k = 4$, and $e_h = 2$ for some $h < k$, change the 3 to 4, the 2 to 0, and stop. Otherwise, change the value of c to 4 (if it was 1). Take e_k as the new c and e_{k-1} as the new d, and return to step (b).

We claim that when BF stops, the 4's are exactly the points of the D-border of C. [The version of BF just given assumes that C is 8-connected and D 4-connected. If the reverse is true, we need only modify step (b) as follows: Let e_k be the first 4-neighbor that has value 1, 3 or 4. If e_{k-1} does not have one of these values, take e_k as the new c and e_{k-1} as the new d. If it does have one of these values, take e_{k-1} as the new c and e_{k-2} as the new d.]

Rosenfeld (1970) showed, by induction on the number of points of S, that BF (or rather, an algorithm closely similar to it) works in the case where $S = C$ is simply-connected. This proof depends on the existence of "deletable points" in any simply-connected C (provided C has more than one point), and so does not generalize directly to an arbitrary C. We can, however, develop a general proof by using the simply-connected case together with a result in (Rosenfeld, 1973) about components that have exactly one hole.

Note first that the operation of BF remains the same if all 0's except those in D and those in the background are changed to 1's. This is because the c's, d's, and e's examined by BF are all in C and D, so that changes to points that lie in neither C nor D can have no effect on BF. (In the case where C is 4-connected and D 8-connected, some of the e's may lie in neither C nor D; but readily, the operation of BF is independent of the values of these e's.) By Proposition 2, it thus suffices to prove that BF works provided C has at most one hole. Note that since the background still consists of 0's, C is still finite.

If C has no holes, it is simply-connected, and the proof in (Rosenfeld, 1970) applies. If C has exactly one hole, by Theorem 4.4 of (Rosenfeld, 1973), it has deletable points unless it is a simple curve (see there for the definitions). But readily, BF works for simple curves; hence here again we can use an induction argument involving deletable points, analogous to the argument used in (Rosenfeld, 1970).

3. The Adjacency Tree

Let B be the background component of $\bar{S}$, and let F, G be distinct components of S or $\bar{S}$. We say that G *surrounds* F, or that F is *inside* G, if any 4-path from any point of F to any point of B must meet G. Readily, B surrounds every other component of S or $\bar{S}$. It is also easily shown that "surrounds" and "inside" are strict partial order relations, i.e.,

(1) if F is inside G, then G is not inside F;

(2) if F is inside G, and G is inside H, then F is inside H.

Our first goal in this section is to prove

THEOREM 3. *Let C be a component of S, and D a component of $\bar{S}$ that is adjacent to C. Then either C surrounds D or D surrounds C; moreover, exactly one such D surrounds C.*

Proof. Let D^* be the component of $\bar{C}$ that contains D (see Proposition 1). Let c^*, d^*, and c, d be pairs of 4-adjacent points in C, D^* and C, D, respectively. If we use BF starting from c^*, d^*, we must eventually obtain c as one of the e's and d as one of the e's, since BF finds the entire D^*-border of C. But readily, the e's having value 0 that are examined by BF are all 4-connected (or, in the modified version of BF, 8-connected); hence d^* is connected to d, so that d^* is in D.

Let D, E be distinct components of $\bar{S}$ that are adjacent to C. By the preceding paragraph, D and E cannot be in the same component of $\bar{C}$. In particular, at most one D_0 can be in the background component of $\bar{C}$; thus every D except for D_0 is inside C. On the other hand, let d_0 be the point to the right of a rightmost point of C; thus d_0 has value 0, and so lies in some D_0, and the right half-line emanating from d_0, which eventually reaches B, never meets C, so that D_0 cannot be inside C. Let ρ be any 4-path from C to B; then ρ meets at least one component of $\bar{S}$ that is adjacent to C, since it must leave C at some border of C. Let D_1 be the last such component that ρ meets; then ρ cannot meet C after leaving D_1, so that D_1 cannot be inside C, which proves that $D_1 = D_0$, and this shows that D_0 surrounds C. ∎

It follows that any C has exactly one *outer border*, namely its D_0-border, where D_0 is the unique adjacent component of $\bar{S}$ that surrounds C; the other D-borders of C, if any, will be called its *inner borders* or *hole borders*. Analogous results hold with S and $\bar{S}$ interchanged, except that the component B of $\bar{S}$ has no outer border.

Let T be the graph whose nodes are the components of S and $\bar{S}$, and in

which two nodes are joined by an arc if the corresponding components are adjacent. As shown above, if D, E are adjacent to C, they cannot be in the same component of $\bar{C}$, i.e., any path joining them must meet C. Thus deletion of node C from T disconnects any two of the neighboring nodes. It follows that T is acyclic, hence is a tree. We shall call T the *adjacency tree* of S. It is convenient to regard node B as the root of T. We can then represent T by a well-formed parenthesis string, with B corresponding to the outermost pair of parentheses.

It should be pointed out that a number of algorithms for finding and marking borders (e.g., Minsky and Papert (1969, pp. 136–139); Rosenfeld (1969, pp. 137–8)) depend on the above results. Specifically, these algorithms make use of the fact that in a TV raster scan of a binary-valued digital picture (i.e., a rectangular array of 0's and 1's, with no 1's on its edges), the outer border of any C is always first encountered as a left–right transition from 0 to 1, while inner borders are first encountered as left–right transitions from 1 to 0. To prove this, let D_0 surround C, and let c be the leftmost of the uppermost points of C; thus the left-hand neighbor d of c is 0. Since the left half-line emanating from d does not meet C, d cannot be inside C; hence d is in D_0. Similarly, let C surround D, and let d be the leftmost of the uppermost points of D; thus the left-hand neighbor c of d is 1. Since the left half-line emanating from c does not meet D, c cannot be inside D; hence c must be in the unique adjacent component of 1's that surround D, i.e., in C.

4. Bicurves

The results in Section 3 were based on the validity of the general BF algorithm of Section 2, which in turn depended on the results of (Rosenfeld, 1970, 1973). In this section we present a more direct method of establishing these results. In what follows, a *pair* (x, y) means a pair of 4-adjacent points, where x has value 1 and y has value 0; we assume, as usual, that the set S of 1's is finite.

We say that two pairs (x, y), (u, v) are *adjacent* if any of the following conditions holds (compare Section 6 of (Rosenfeld, 1970)).

(a) x is 4-adjacent to u and y is 4-adjacent to v.

(b) $v = y$ and x is 8-adjacent to u.

(c) $u = x$, and y is 8-adjacent to v, and the other common 4-neighbor of y and v has value 0.

Thus (x, y) and (u, v) are adjacent in the following cases.

$$\begin{matrix} x & u; \\ y & v; \end{matrix} \qquad \begin{matrix} x & & w \\ y & = v & u \end{matrix} \quad \text{whether } w = 0 \text{ or } 1; \qquad \begin{matrix} x = u & v. \\ y & 0. \end{matrix}$$

This definition is designed for the case where we use 8-connectedness for the 1's and 4-connectedness for the 0's. To handle the opposite case, we need only assume that first terms of pairs are 0's and second terms 1's, and replace "0" by "1" in (c).

PROPOSITION 4. *Any pair has exactly two adjacent pairs, one in each of the two-by-two squares that contain the given pair.*

A sequence of distinct pairs $(x_1, y_1),\ldots, (x_n, y_n)$ will be called a *bipath* if (x_i, y_i) is adjacent to (x_{i+1}, y_{i+1}), $1 \leqslant i < n$. A bipath will be called a *bicurve* if $n > 2$ and (x_n, y_n) is adjacent to (x_1, y_1). By Proposition 4, in any bicurve, the only pairs adjacent to (x_i, y_i) are the two pairs $(x_{i\pm1}, y_{i\pm1})$(modulo n). Note also that the 1's in two adjacent pairs are always 8-connected, and the 0's always 4-connected; hence the first terms of the pairs in any bipath are 8-connected, and the second terms 4-connected.

PROPOSITION 5. *Distinct bicurves are disjoint.*

PROPOSITION 6. *Let P be a set of pairs such that, for all $(x, y) \in P$, exactly two pairs of P are adjacent to (x, y). Then P is a union of bicurves.*

Let β be a bicurve, p any point, and let H_p be the right half-line emanating from p. We say that H_p *crosses* β at (x_i, y_i), where (x_i, y_i) is a pair in β, if x_i and y_i both lie on H_p. We say that p is *inside* β if the number of times H_p crosses β is odd; *outside*, if the number is even.

PROPOSITION 7. *If p and q are 4-adjacent and are not a pair in β, or if they are 8-adjacent* 1's, *they are either both inside or both outside β. If they are a pair in β, one of them is inside and the other outside.*

Proof. This is clear if p and q are horizontally adjacent. Suppose they are vertically adjacent and are not a pair in β. Let $p = (x_r, y_r),\ldots, (x_s, y_s)$ be a run of pairs in β both of whose terms lie on $H_p \cup H_q$; thus both terms of (x_{r-1}, y_{r-1}) and (x_{s+1}, y_{s+1}) do not lie on $H_p \cup H_q$. It is easily verified that for any such run, either H_p crosses β twice and H_q does not cross it; or vice versa; or both H_p and H_q cross β once. Thus in any case, the difference between the numbers of crossings is even, and since this is true for any run, p and q are either both inside or both outside β.

A similar argument applies if p and q are diagonally adjacent 1's; one must verify here that no trouble can arise in the case of a run just at the beginning of $H_p \cup H_q$. Note that if p and q are diagonally adjacent 0's, a run can cross H_p at its beginning without crossing H_q, e.g., for $\begin{smallmatrix} p & u & z \\ x & q & w \end{smallmatrix}$, where $x = u = 1$, $z = w = 0$, we can have the pairs (x, q), (u, q), (u, z).

Finally, if p and q are a pair in β, there is in fact a run just at the beginning of $H_p \cup H_q$, namely the run containing (p, q); and this run crosses either H_p or H_q, but not both. All other runs still give even differences in the numbers of crossings; hence the total numbers have opposite parity, so that p, q cannot be both inside or both outside. ∎

COROLLARY 8. *If p and q are 4-connected 0's or 8-connected 1's, they are either both inside or both outside β.*

COROLLARY 9. *The background 4-component B of 0's is outside β.*

COROLLARY 10. *Let p be inside β, and let $t_1, \ldots, t_n$ be a 4-path from p to a point of B; then (t_i, t_{i+1}) is a pair in β for some $1 \leqslant i < n$.*

Let C be any component of S, and D any component of $\bar{S}$ that is adjacent to C. Let (C, D) be the set of pairs (x, y) with $x \in C$, $y \in D$. It is easily verified that (C, D) has the property of Proposition 6, hence is a union of bicurves. Let (u, v) be a rightmost pair in (C, D), i.e., no term of any pair in (C, D) is further to the right than u or v. Suppose that (u, v) is a pair in the bicurve β of (C, D). Readily, u and v must be horizontally adjacent. Thus if u is to the left of v, C is inside β and D outside, and if v is to the left of u, the reverse is true. Moreover, let β_1 be any other bicurve of (C, D), and let (c_1, d_1) be a pair in β_1. Then c_1 is inside β and d_1 outside (or vice versa); but (c_1, d_1) isn't a pair in β (Proposition 5), which contradicts Proposition 7. We have thus shown that (C, D) is a single bicurve, with C inside it and D outside it (or vice versa). It follows from Corollary 10 that C is inside D (or vice versa).

Finally, let C be inside both D and E; let (c', d) and (c'', e) be pairs in (C, D) and (C, E), respectively. Evidently these bicurves are distinct, hence disjoint; thus by Proposition 7, d is inside (C, E) and e inside (C, D), so that by Corollaries 8 and 10, D is inside E and E inside D, contradiction. Hence at most one D adjacent to C can have C inside it; all other such D's must be inside C.

Evidently, if D is inside C and C is inside E, any path from D to E must meet C. Let D and E both be inside C; then E is outside (C, D), while D is inside it, so that any path from D to E must cross (C, D)(in the sense of

Corollary 10), hence must meet C. Using these results together with the analogous results with 1's and 0's interchanged, we thus have once again proved that the adjacency graph T is a tree.

5. Construction of the Adjacency Tree

Suppose that we divide the background component B into two connected parts B_1 and B_2, where B_1 is finite and surrounds the other components to which B was adjacent, and B_2 surrounds B_1. The adjacency tree of components, with B_1 replacing B and B_2 excluded, is unchanged; and we can now describe algorithms for constructing this tree by scanning the finite set $I \times I - B_2$. Note that in particular, if we take B_2 to be the outside of a large rectangle, $I \times I - B_2$ is just a binary-valued digital picture (i.e., a rectangular array of 0's and 1's) with 0's outside it.

Our first (well-known) method of constructing the tree depends for its validity on the following observations: If C is inside D and has no holes, for any x in C there exists y on the D-border of C, to the left of x, with nothing but 1's between y and x; for, as we move left from x, the 1 that we pass just before we first hit a 0 is a border point of C, and this can only be on the D-border. Conversely, if C has holes, there exists a point z on the C-border of some hole, and y on the D-border of C to the left of z, with nothing but 1's between y and z. Indeed, the lefthand neighbor z of a leftmost hole point is in C, and there can be nothing but 1's between y and z. The above assumes that C is a component of 1's; the analogous observations also hold with 1's and 0's interchanged.

To construct the tree, we start at a point of the outer border of B_1. We uniquely mark this starting point, and use BF to mark the entire border, say with stars. We then use BF again, but this time, after we move to each border point b, we also scan to the right through 0's or primed points inside B_1, marking any nonprimed 0's with primes as we go, until we hit a starred point. We then move leftward back to b (which is identifiable, since it is starred) and resume BF. If we can finish this process and return to the starting point, we can conclude by the remarks in the preceding paragraph that B_1 has no holes. Thus in this case, the tree consists of a single pair of parentheses. On the other hand, if B_1 has holes, we will hit a 1 at some stage before hitting a starred point; this 1 is on the B_1-border of some component C of 1's adjacent to B_1.

Suppose, in general, that we have already constructed the tree for all components outside C, and that we have just found C by hitting a point of its

outside border, as in the preceding paragraph. The "current" pair of parentheses π in the string being constructed represented the component (D, say) adjacent to and surrounding C; when we hit C, we add a new pair π' inside π, representing C itself, and π' becomes the new current pair. We then treat C exactly the way we treated B_1 in the preceding paragraph (but with 1's and 0's interchanged, if C is a component of 1's). If we find a hole E in C, we add a new current pair of parentheses inside π', and repeat the process of the preceding paragraph. If not, we can finish the scan of C, so that it has no holes, and π' is an innermost pair of parentheses in the string. In this case we once again use BF to traverse the outer border of C and turn all stars into primes; thus every point of C has now been primed. We now revert to π as the current pair of parentheses, and resume scanning D. If D has other holes that have not yet been primed, this scan will find them; if not, we will be able to finish scanning D and resume scanning its surrounding component. Ultimately, this process will enable us to finish scanning B_1, at which point the procedure terminates.

We conclude by describing a second method of constructing the adjacency tree; this is a simplified version of a procedure due to (Bunemann, 1969). Here we shall assume that we are given a rectangular array of 0's and 1's with no 1's at its edges; i.e., the top and bottom rows, and leftmost and rightmost columns, are all 0's, all of which belong to the background component B. In the process of constructing a parenthesis-string representation of the adjacency tree, we will use brackets instead of parentheses to represent components that are currently being tracked. We assume that we have some way of telling, for any run of 1's or 0's on the current row of the array, to which pair of brackets it corresponds.

We scan the top row of the array and construct an initial string consisting of a single pair of brackets, corresponding to the background region. Suppose that we have scanned the first k rows, and have constructed a string of brackets and parentheses representing the adjacency tree for the subarray consisting of these k rows; we can certainly do so for $k = 1$. We now scan the $(k + 1)$ st row and update the string as follows.

(a) If a component C, say of 1's, that met the kth row does not meet the $(k + 1)$ st row, i.e., none of C's runs on the kth row is adjacent to any run of 1's on the $(k + 1)$ st row, we turn C's brackets into parentheses. In particular, if the kth row was the bottom row, the outermost pair of brackets turns into parentheses; by our assumption that only the background component meets the bottom row, this is the only pair of brackets in the string at that stage.

(b) If a component C, say of 1's, that met the kth row, splits on the $(k + 1)$ st row, i.e., one of C's runs is adjacent to two or more runs of 1's on the $(k + 1)$ st row, we insert one or more new pairs of brackets inside C's brackets, corresponding to the new runs (of 0's) on the $(k + 1)$ st row, each of which may be the beginning of a hole in C.

(c) If two (or more) components D, E, say of 1's, that met the kth row merge on the $(k + 1)$ st row, i.e., a run of 1's on the $(k + 1)$ st row is adjacent to runs belonging to both D and E, we combine their brackets into a single pair; the substrings (if any) that were inside their brackets are concatenated and are put immediately inside this new pair of brackets.

It is not hard to see that the resulting updated string correctly represents the adjacency tree for the subarray consisting of the first $k + 1$ rows.

RECEIVED: December 4, 1973

REFERENCES

BUNEMAN, O. P. (1969), A grammar for the topological analysis of plane figures, *in* "Machine Intelligence 5" (B. Meltzer and D. Michie, Eds.), pp. 383–393, Edinburgh University Press.

MINSKY, M. AND PAPERT, S. (1969), "Perceptrons," M.I.T. Press, Cambridge, Mass.

ROSENFELD, A. (1969), "Picture Processing by Computer," Academic Press, New York.

ROSENFELD, A. (1970), Connectivity in digital pictures, *J. Assoc. Comput. Mach.* **17**, 146–160.

ROSENFELD, A. (1973), Arcs and curves in digital pictures, *J. Assoc. Comput. Mach.* **20**, 81–87.